notiser och
rapporter från

PEDAGOGISK-
PSYKOLOGISKA
INSTITUTIONEN

LÄRARHÖGSKOLAN
FACK, 200 45 MALMÖ 23

pedagogisk-psykologiska problem

Robertsson, L.:

DATORPROGRAM TILL ANACONDA – EN TEKNIK FÖR TEXTANALYS

Nr 317 April 1977

DATORPROGRAM TILL ANACONDA - EN TEKNIK FÖR TEXTANALYS

Leif Robertsson

Robertsson, L. Datorprogram till ANACONDA - en teknik för textanalys. Pedagogisk-psykologiska problem, (Malmö: Lärarhögskolan), Nr 317, 1977.

I denna rapport presenteras en översiktlig beskrivning av de metoder för datamaskinell bearbetning som använts för utprövning av ANACONDA (Analys av concept genom datorbearbetning). Särskild vikt lägges vid det program som utvecklats för identifiering av ord i texten med ord i lexikon. I form av en bilaga redovisas bakgrundsinformation om ANACONDA-metoden.

Nyckelord: Kvantitativ lingvistik, datorprogram, intervjudata, psykolingvistik.

INNEHÅLL Sid

1. NÅGRA KOMMENTARER

Beteendevetenskaplig forskning kräver inte bara att forskare kan bearbeta och analysera data i numerisk form utan också data som föreligger i form av text. Olika textmaterial är ofta mycket omfattande, vilket gör att omsorgsfulla, objektiva, reliabla och valida analyser blir mycket betungande och tidskrävande moment i en forskningsprocess. Men trots såväl många teoretiska som metodiska och tekniska eller praktiska problem utförs nästan inom varje större forskningsprojekt innehållsanalyser av textmaterial.

Betydelsen av en innehållsanalys och de forskningsmetodiska krav som måste ställas kan naturligtvis variera mycket kraftigt från fall till fall. Vissa forskningsproblem lämpar sig till exempel väl för insamling av information i huvudsakligen numerisk form. I sådana fall kan det vara tillräckligt med en impressionistisk analys av verbala utsagor. Föreligger emellertid ett forskningsproblem som kräver att datainsamlingen sker med hjälp av intervjumetoden så leder detta till huvudsakligen verbala data. I ett sådant fall blir innehållsanalysmetoden huvudmetoden för dataanlysen, med betydligt större forskningsmetodiska krav som följd.

Ett forskningsprojekt, kallat Skolpedagogiska sökstrategier (SÖK) som finansierats av Skolöverstyrelsen, utfördes i syfte att kartlägga hur forskare på Sveriges pedagogiska institutioner ser på (1) problemperception och problemformulering och (2) informationssökning och spridning. Mot bakgrund av problemställningen bedömdes intervjumetoden som enda lämplig datainsamlingsmetod. De intervjuer som utfördes med 40 slumpvis valda forskare resulterade i 4000 sidor text. Materialets karaktär och omfång ledde till att vi skulle försöka använda oss av datorkraft i våra analysförsök. Med tanke på datorns bearbetningshastighet, flexibilitet, exakthet, outtröttlighet och tillförlitlighet har vi utvecklat ett system för en datorbaserad innehållsanalys. Vår metod för en analys av concept genom datorbearbetning kallar vi ANACONDA. Ett knippe ledtrådar till ANACONDA-systemet presenteras som bilaga.

Datorns höga bearbetningshastighet gör det möjligt att stora mängder av verbala data kan bearbetas. Men en datorbaserad bearbetning av text kräver också att algoritmer kan utvecklas och datorprogram kan skrivas som

1. accepterar den struktur som kännetecknar ett naturligt språk
2. systematiskt identifierar lingvistiska tecken och strängar som förekommer ensamma eller gemensamma med andra tecken eller strängar i en text
3. sorterar lingvistiska element
4. reorganiserar lingvistiska element i överensstämmelse med en viss bestämd syntaktisk position

5. utför logiska selektioner
6. räknar frekvenser och skriver ut frekvensstatistik i form av matriser.

Datorns stora flexibilitet innebär att en uppgift kan uttryckas med olika instruktioner på samma sätt som en och samma tanke kan uttryckas med hjälp av olika syntaktiska konstruktioner. Men detta betyder samtidigt också att ett program kan vara skrivet med olika krav på "elegans".

Det program som har utvecklats för utprövning av ANACONDA har i första hand skrivits med syftet att få metoden att fungera och att uppnå tillfredsställande resultat vid bearbetningen av ett empiriskt material. Av detta skäl har det varit av underordnad betydelse att åstadkomma generaliserade rutiner och den "elegans" som programmeraren skulle önska.

Bernhard Bierschenk
Malmö, vt 1977

2. NÅGRA PROGRAMMERINGSTEKNISKA ASPEKTER

Programutvecklingen för ANACONDA och den maskinella bearbetningen av intervjumaterialet har skett vid Lunds Datacentral (LDC). LDC fungerar som ett serviceorgan för forskning och utbildning. Genom sina programmerare ger den hjälp vid utveckling och skrivning av program. Det programutvecklingsarbete som presenteras i denna rapport har utförts som stödprogrammering.

2.1 Maskinell utrustning

Den maskinella databehandlingen har utförts med UNIVAC 1108. Centralanläggningen innefattar

1. Centralenhet med 0.75 μ
2. 196 000 ord som kärnminne (ordlängd 36 bitar)
3. Tre kortläsare med läshastigheten 600 kort i minuten vardera
4. Tre radskrivare med skrivhastigheten 600 kort i minuten vardera
5. Snabbradskrivare med kapaciteten 1600 rader i minuten
6. Skivminne med en minneskapacitet av 600 miljoner tecken
7. Sex magnetbandstationer
8. Annan utrustning som ej använts i samband med programutvecklingen för ANACONDA

Datorns arbete kontrolleras av operativsystemet EXEC 8. Användaren kommunicerar med detta med hjälp av ett styrspråk med samma namn. Styrspråket innehåller bl a direktiv för inläsning, korrektion och kopiering av datafiler på olika media. Ett enskilt användarprogram får inte utan särskilda skäl belägga mer än ca 62 000 helord i kärnminnet.

Det finns kompilatorer för bl a Algol och FORTRAN, de högnivåspråk som använts i projektet. Programbiblioteket är omfattande. För beteendevetare med statistiska tillämpningar finns bl a Statistical package for the social sciences (SPSS), Biomedical computer programs (BMD, BMDP) och International mathematical and statistical libraries (IMSL).

2.2 Inläsning av huvudmaterialet

Inläsning av det stansade materialet har följt en vanlig procedur. Inläsningen initieras av en styrspråksinstruktion som stansas på hålkort och läggs främst i datamaterialet. Samtliga hålkort matas sedan i en kortläsare och innehållet kopieras då till maskinens massminne. På massminnet utgör det nu en datafil, som är lagrad på ett standardmässigt sätt i s k SDF-format (System Data Format). Därmed är det lätt tillgängligt för hantering i styrspråket och olika pro-

gramspråk. Lagring på massminne är ganska dyrbart och när filen inte skall användas för bearbetningar på viss tid kopieras den till magnetband och raderas ut från massminnet. Vid förnyad användning kopieras filen sedan tillbaka till massminnet. Detta är moment som programmeraren själv får initiera i enkla körningar.

2.3 Kontroll av materialet

För ett material av den aktuella storleksordningen (och även mindre) är det rimligt att anta att det måste finnas fel i materialet. Fel som innebär brott mot givna kodningsregler kan diagnostiseras med hjälp av datamaskinen, om man gör ett program som läser datafilen och kontrollerar innehållet i datakorten med avseende på sådana regler. Kontroll och rättelser görs ofta i två steg. Det första steget kontrollerar att kortordningen är den riktiga för objekt som måste beskrivas på flera kort. I vårt fall har ett särskilt program skrivits i FORTRAN, som kontrollerar att individerna finns i löpande ordning 1-40, frågorna i löpande ordning 5-8, meningarna i löpande ordning från 1 och uppåt inom meningarna. En särskild komplikation att ta hänsyn till visade sig vara ett par förekomster av icke-numeriska tecken (felstansningar) i kontrollkolumnerna, som fick den första versionen av programmet att spåra ur.

Programmet gav som resultat en lista med de kort utskrivna vid vilka hopp eller dubbla förekomster fanns i den löpande numreringen. Bl a visade det sig att för en intervjuperson samtliga data fanns i dubbel upplaga. Utskrifterna markerades med kortens ordningsnummer i filen för att underlätta rättningen.

Efter korrigeringar så att kortens numrering i filen är korrekt, sker det andra steget i kontroll- och rättelseproceduren. Då kontrolleras det egentliga innehållet i varje kort för sig. I det aktuella fallet gjordes en utskrift av hela materialet på radskrivarpapper (ca 600 sidor), och därefter en manuell genomgång. Om innehållet i en textkolumn är fel, så kan detta ofta endast avgöras av en person med kännedom om texten. Däremot kunde i efterhand konstateras att satskoderna hade bort kontrolleras maskinellt på detta stadium. Vissa fel här kvarstod efter genomgången och uppdagades på ett senare stadium, då materialet började att sorteras på olika koder.

2.4 Rättning av materialet

I styrspråket för UNIVAC 1108 finns uttrycksmedel för att från en gammal fil skapa en ny, rättad, genom att skriva en följd av rättelsekort som innehåller radnummer och en kod för sättet att ändra i raden samt de rättelser som skall göras i raden. En eller flera på varandra följande rader kan tas bort eller ersättas med en följd av nya. Nya rader kan läggas in efter en angiven rad. På

en angiven rad kan en följd av särskilt angivna kolumner ändras. Denna metodik har använts vid framställning av korrektioner till materialet. Efter inläsning av korrektionerna har ovan beskrivna kontrollprogram körts på nytt. I själva verket har då några fel kvarstått och rättelseproceduren har fått upprepas.

2.5 Sortering och sorterade utskrifter

I de inledande analyserna av materialet har gjorts utskrifter av flera olika urval av de enskilda kodade orden. Orden har i utskrifterna varit sorterade på olika sätt enligt önskemål. I många av de program som framställs har förutsatts att de datafiler som programmet använder är sorterade på ett visst sätt. Dessa sorteringar har gjorts med hjälp av ett generellt sorterings- och konverteringsprogram, TESTST. Detta skrevs åren 1970-72 av Peter Bergh vid Lunds Datacentral. UNIVAC:s egen programvara för sortering har ansetts som mindre flexibel och effektiv.

Programmet skapar från en angiven fil en ny, sorterad fil. Sorteringsordningen bestäms av ett antal nycklar i hierarkisk ordning. En nyckel består av angivandet av en följd av närliggande kolumner i datakortet. En uppgift som man sorterar efter måste således finnas i samma kolumner i varje kort. I vårt fall har vi emellertid ibland önskat sortera efter t ex satskoder, som står i olika kolumner, beroende på om satsen är huvudsats eller bisats. Då har man först fått göra ett enkelt redigeringsprogram som tillfälligt skapar en fil där satskoden placerats i en särskild kolumn vid sidan av de egentliga uppgifterna i kortet, och sedan har denna fil sorterats. Många liknande små program har gjorts under projektets gång och skrivits i Algol eller FORTRAN.

3. FRAMSTÄLLNING OCH BEARBETNING AV ETT SKATTNINGSFORMULÄR

Utgångspunkt för framställningen av ett skattningsformulär var en lista med adjektiv (N = 570) och en med verb (N = 883). Dessa ord skulle presenteras 16 bedömare för att skaleras enligt tre olika dimensioner. Med datamaskinens hjälp skulle framställas för varje bedömare och dimension en utskrift av orden i en ny, slumpmässig ordning. Till höger om ett ord skulle skrivas ut sifferföljden 1-7 för att markera den 7-gradiga skattningsskalan. Bedömare skulle sedan sätta en ring kring den siffra som markerar hans skattning av ordet.

Ett program för uppgiften har skrivits i FORTRAN. Programmet är ganska kort och utnyttjar ett par standardfunktioner, RANF och QSORT, som inte är FORTRAN:s standard men som finns vid Lunds Datacentral. RANF genererar rektangulärfördelade slumptal i intervallet (0, 1). QSORT sorterar talföljder internt i kärnminnet. Programmet arbetar i följande steg:

1. Läs in en ordlista och lagra i ett fält. Orden förutsätts sorterade i bokstavsordning. (Antal ord = N.)
2. Skriv ut en första sida som rubricerar ordlistan och anger ordtyp (adjektiv/verb), bedömarens nummer, dimension
3. Bilda en serie av N slumptal och anknyt dem till deras ordningstal 1, 2, ... N som anger den ordning i vilken slumptalen bildades
4. Ordningstalen sorteras om enligt slumptalens storleksordning och vi får då en slumpmässig serie av ordningstal
5. En ordlista skrivs ut. Det första ordet hämtas från den placering i ordlistan som det första ordningstalet anger, osv
   På varje sida skrivs två kolumner om 30 ord vardera. Varje sida inleds med en sex-siffrig sididentitet med ordtyp, kod för dimension och bedömare samt sidnummer
6. Följden av ordningstal skrivs ut på massminne för att sparas tills formulären fyllts i och bearbetats i maskinen. Skattningarna på en sida kan då helt enkelt stansas på ett hålkort med sididentitet och 60 inringade tal. Datamaskinen "minns" vilket ord som står i ett visst läge på varje sida
7. Punkt 2-6 upprepas 16 x 3 = 48 gånger

Ett exempel på programmets arbetssätt redovisas i ruta 1.

Maskinen hämtar från massminnet uppgiften att den första kodsiffran i steg 5, nämligen skattningsvärdet sex (6), tillhör ordet "sitta", dvs ord nr 3 i den ursprungliga listan (se steg 1). Det andra skattningsvärdet (5) tillhör ordet "gå",

Ruta 1. Exempel på en datamaskinell framställning och bearbetning av skattningsformulär

Steg 1: N = 5

1. gå
2. ligga
3. sitta
4. springa
5. stå

Steg 2:

VERB
Aspekt = 1
Individ = 1

Steg 3:

0.263	1
0.671	2
0.131	3
0.444	4
0.918	5

Steg 4 & 6:

0.131	3
0.263	1
0.444	4
0.671	2
0.918	5

sparas i massminne

Steg 5:

Utskrift
210101 (sidnr)

	neg						pos
sitta	1	2	3	4	5	(6)	7
gå	1	2	3	4	(5)	6	7
springa	1	2	3	4	(5)	6	7
ligga	1	2	3	(4)	5	6	7
stå	1	2	(3)	4	5	6	7

Inringade siffror anger skattningar. Den skattade sidan stansas:
210101 65543

dvs ord nr 1 i den ursprungliga listan. På detta sätt fortsätter tillordningsprocessen. Ett program har skrivits i FORTRAN som tar hand om alla skattningar. Resultatet blir den ursprungliga ordlistan med en kortbild per ord, nu kompletterad med de 15 x 3 skalvärdena (en bedömare utgick) som givits för ordet.

Skattningar har sedan beskrivits med hjälp av dels egna, mindre program, dels program som är tillgängliga i t ex BMDP och SPSS. Ordlistan omformades sedan än en gång till att innehålla kortbilder med ord, samt ett medelvärde per dimension över bedömarnas skattningar.

4. PROGRAM FÖR INFORMATIONSSÖKNING OCH -ÅTERVINNING

Det program som nedan skall beskrivas har varit det mest komplicerade enskilda programmet som utvecklats för ANACONDA. Givna har varit tre lexikon för adjektiv, verb respektive substantiv. En rad i ett lexikon kan innehålla ett ord eller en ordföljd. Ett ord kan vara trunkerat och ett eller flera ord i ordföljden kan vara trunkerade. Givna har också varit tre listor med ändelser för adjektiv, verb och substantiv. Datorns uppgift är nu att identifiera ord eller ordföljder som förekommer med viss satskod i huvudregistret med en rad i motsvarande lexikon. Om ett ords början stämmer överens med ett trunkerat ord i lexikon, så måste fortsättningen av ordet vara en tillåten ändelse för att identifikation skall ske.

4.1 Lexikonstruktur

En rad i ett lexikon kodades för hålkortsstansning efter följande enkla regler:

1. Det första ordet börjar i kolumn 1
2. Trunkering markeras med en asterisk (*) omedelbart efter ordet
3. Ett ord eller en ordföljd som saknar trunkeringar avslutas med ett blanktecken följt av likhetstecken. (Denna kod är egentligen överflödig.)

I ruta 2 presenteras ett exempel på trunkeringsprincipen.

Ruta 2. Exempel på ord i lexikon

```
HÖLL  MIG TILL =
VISA*
VISA* SIG
```

Trunkerade ord som inte står sist i raden leder till särskilda problem. Dessa innebär bl a att ett lexikon bör presenteras för maskinen med raderna i en lämplig sorteringsordning och med orden och trunkeringarna beskrivna i en kvantifierad form. Ett första steg efter inläsning var därför att sortera kortbilderna i alfabetisk ordning men så att asterisk och blanktecken kommer före alfabetet och särskilt asterisk före blanktecken i sorteringsordningen. Identifiering av textavsnittet med en rad i lexikonet måste ske genom en systematisk genomgång av detta. Proceduren kompliceras av att ett textavsnitt som har en motsvarighet i lexikonet kan innehållas i ett annat som också kan identifieras där. I detta fall skall det längre avsnittet identifieras med texten. Förhållandet beror på att det finns ord i lexikonen som innehålls i andra ordföljder, eventuellt efter att trunkering ersatts med ändelse. Den angivna sorteringsordningen

innebär att ett ord som innehålls i ett annat kommer före detta. Om det nu görs en fullständig genomgång av lexikonet från början till slut så kan det bli flera utfall för identitet och det blir då det sista som skall gälla. Ett exempel på ordidentifiering presenteras i ruta 3.

Ruta 3. Exempel på ordidentifiering

I texten står		I lexikonet står
HÅRT PRESSAD		HÅR* HÅRT PRESSA*
Först identifieras		
HÅRT	med	HÅR*
Därefter		
HÅRT PRESSAD	med	HÅRT PRESSA*

Det senare ordet i lexikonet identifierar textavsnittet. Den i ruta 3 presenterade principen används i identifieringsprogrammet. Proceduren är emellertid tidsödande varför vi försökte att utveckla en mera effektiv rutin. Detta skedde genom en särskild klassificering av ord. Förekomsten av de trunkerade orden medför att den s k binärsökningsmetoden inte utan vidare kan tillämpas som alternativ sökprocedur.

Innan lexikonen läses in i maskinen omkodas de på följande sätt. De två första tecknen i raden anger ordföljdens längd fram till och med första trunkerade ord eller till ordföljdens slut. Det tredje tecknet anger med kodsiffran 1 om denna del av ordföljden inte är trunkerad, annars 0. Det fjärde tecknet anger med kodsiffran 1 om ordföljden fortsätter efter det trunkerade ordet, annars 0. Från och med det femte tecknet lagras den aktuella delen av ordföljden. Om det finns fortsättning, så kommer omedelbart därefter en fyrsiffrig kod, som beskriver nästa delordföljd fram till nästa trunkering samt texten. Proceduren kan upprepas så länge det behövs. Proceduren exemplifieras i ruta 4.

Ruta 4. Exempel på omkodning av ord

ABONNERA*	800ABONNERA
FÖRSTOD	710FÖRSTOD
HA* NYTTA AV	201HA 810NYTTA AV

Den omkodning som exemplifieras i ruta 4 har utförts med ett separat Algol-program.

4.2 NU - Algol

Ett informationssökningsprogram har skrivits i den på UNIVAC 1108 fungerande Algol-versionen NU (Norwegian University)-Algol. Detta språk har som kärna Algol-60 men har utvidgats till att omfatta stränghantering (en sträng är liktydig med en följd av tecken). Det är denna utvidgning som gjort språket användbart i vårt fall. Således har införts en ny typ för variabler:

"string" (sträng) (1)

Strängar deklareras med variabelnamn och längd enligt följande exempel:

string k (84), s (3); (2)

Det finns också indicerade strängvariabler som deklareras såsom fält av strängar av given längd med undre och övre indexgränser angivna. Detta kan exemplifieras på följande sätt:

string array adj (36:1:586); (3)

I "adj" kan lagras 586 ord om max 36 tecken. Med adj (i) kommer vi åt det i-te ordet.

Med hjälp av programmet kan vi hantera delsträngar genom att ange begynnelseposition och antal tecken i en sträng. Med uttrycket

k (12, 2) (4)

kan vi t ex återvinna tecken nr 12 och nr 13 i sträng k. Med det allmänna uttrycket k (i) kan vi återvinna det enskilda tecknet med nr (i) i sträng k.

Programmet accepterar strängkonstanter som består av en följd av tecken inneslutna av enkla anföringstecken. Strängar och strängkonstanter kan lagras i strängvariabler enligt följande exempel:

s: = ′abc′; k (36, 3): = s; adj (500): = s; (5)

Strängar kan jämföras med varandra med avseende på likhet eller sorteringsföljd. Detta uttrycks på följande sätt:

if k (i) = ′(′ then (6)
if s > ′maa′ then (7)

Strängar med rent numeriskt innehåll kan konverteras till och från heltalsvariabler. NU-Algol innefattar möjligheten att anropa separat kompilerade FORTRAN-subrutiner. Denna möjlighet har utnyttjats i sökprogrammet i ett avseende. NU-Algols egna uttrycksmedel för hantering av massminnesfiler i

SDF-format är primitiva och det har varit bekvämare att dirigera sådan in- och utmatning via FORTRAN-subrutiner.

4.3 Binärsökning på begynnelsebigram

Sökprogrammet börjar med att läsa in adjektiv-, verb- och substantivlexikonet till kärnminnet. De hämtas från massminnet i den sorterade och omredigerade form som beskrivits ovan. Varje lexikon lagras i ett fält av strängar (string array). Därmed har det längsta ordet genom manuell kontroll i förväg fått bestämma strängarnas längd, så att adjektiv och verb fått uppta 36 tecken, substantiv hela 66 tecken. Vi offrar med detta en del kärnminnesutrymme för annan bekvämlighet. Med större lexikon kunde kärnminnesutrymmet blivit kritiskt och lagringssättet måst modifieras. Nu upptar själva orden 31 132 helord. Verb- och substantivlexikonen upptar över 1 600 ord vardera.

Den angivna sökmetoden innebär egentligen att samtliga ord i lexikonen måste jämföras med det aktuella ordet i texten. Detta skulle leda till orimligt långa körtider, och processen måste snabbas upp. Vi har här använt en enkel mekanism, som bygger på att inget ord i lexikon trunkeras före det tredje tecknet. Ett lexikon kan då anses som uppbyggt av ett antal segment, där orden inom ett segment börjar på samma bigram (kombination av två tecken). I inläsningsproceduren har därför lagts in ett moment som skapar en referenslista med en följd av variabelpar. Dessa innehåller varje nytt bigram som lästs in samt ordningstalet i lexikon för det första ordet som börjar med detta bigram. I själva sökprocessen identifieras sedan det aktuella textordets två första bokstäver med ett bigram i referenslistan medelst binärsökning. I referenslistan har bigrammet en motsvarande begynnelseadress i lexikon och sökning efter det fullständiga ordet kan ske i ett begränsat område. Det visade sig att adjektiven börjar med ca 180 olika bigram, verben med ca 170 och substantiven med ca 240.

De särskilda ändelseregistren är av mindre omfattning och läses in direkt som sorterade kortbilder till vardera ett fält av strängar om 6 tecken.

4.4 Hantering av satsstrukturer

Textmängden finns lagrad på massminne. Den största enskilda mängden text som behöver bearbetas samtidigt är en mening. Vid bearbetningstillfället hämtas från massminnet en mening åt gången och läses in i kärnminnet. Meningen lagras i ett fält av strängar om 84 (80 relevanta) tecken. Fältets längd har maximerats till 100 kortbilder.

En mening består av en huvudsats samt högst fyra bisatser. Satserna identifieras på grundval av satskodernas placering i fem olika kolumner till höger

i kortbilderna, och utgör i detta sammanhang logiskt skilda enheter. Därför gör programmet fem genomgångar av en mening och hanterar vid varje genomgång rader i texten som markeras i en särskild kolumn för satskod.

Innan en rad i texten bearbetas flyttas den till ett särskilt buffert-utrymme med plats för ca 200 tecken. Det finns flera anledningar till detta. En rad i texten kan vara så lång att den finns kompletterad i särskilda fortsättningskort och den skarvning som måste företagas i programmet förenklas i ett initialskede.

Varje sats har i princip högst ett verb. I texten finns ändå ofta flera rader i samma sats där satskoden anger verb, och det kan finnas andra rader emellan. Så är det ofta när verbet består av hjälpverb och huvudverb. Programmet sätter samman dessa olika verbkomponenter till en följd av ord med ett blanktecken emellan för att få representationen jämförbar med lexikonen och lagringen sker i buffert-utrymmet. Det förutsätter då att ordföljden är rak.

Flyttningen av en rad i texten tar en viss tid i anspråk. I gengäld snabbas den egentliga bearbetningen upp genom att enskilda tecken hämtas från en enkel strängvariabel istället för en indicerad sådan. Vid flyttningen beräknas också den egentliga textradens längd, en uppgift som används i bearbetningen.

4.5 Hantering av satskoder

Vid identifieringen av en enskild rad i texten används satskoder. I ruta 5 presenteras jämförelsen av en rad i texten med respektive lexikon.

Ruta 5. Identifiering av enskilda textrader

Adjektivlexikon om satskoden är	32, 52 eller 72
Verblexikon om satskoden är	40
Substantivlexikon om satskoden är	30, 33, 50, 53, 70, 73, 44, 45, 46

Efter klassificering av rader efter satskod som sker i huvudprogrammet, anropas en procedur (subrutin) som skall ge det identifierade ordet i lexikon om sådant finns och som initieras med information om vilket lexikon och vilka referenstabeller som är de aktuella.

En rad kan leda till flera genomsökningar av lexikon, om den innehåller flera enskilda ord. Första sökningen sker med det första ordet som inledning till den text som skall identifieras. Om ingen identifiering sker upprepas sökningen med nästa ord som inledning till texten. Om identifiering sker startar en ny sökning med början från det första ord som inte ingår i det identifierade textavsnittet. Anledningen till detta är att flera identifierbara ord kan finnas i en

rad. En konvention har då införts att det sista ordet är det väsentligaste och skall vara det som slutgiltigt identifieras. Ett undantag är de supplerade orden. I raden kan finnas ordföljder som inleds och avslutas av vänster- och högerparenteser. Dessa sekvenser avbryter den egentliga textföljden och hanteras på samma sätt. Supplerade ord som identifieras prioriteras före ord i den egentliga texten. Proceduren har således följande uppgifter:

1. Stegvis prövning av nya textavsnitt i en rad med början i nya ord
2. Binärsökning i referenstabell efter den aktuella textens inledande bigram
3. Prövning av samtliga ord i lexikon som börjar på samma bigram
4. Prioritering av text som ingår i supplering

Prövning av ett ord i lexikon enligt punkt 3 innebär i sig självt ett ganska omfattande programsteg. Det är här som trunkeringarna måste beaktas. Första delen av ett ord i texten måste stämma överens med det trunkerade ordet och dessutom måste resten av ordet vara en tillåten ändelse. Vid jämförelse av ändelsen med ändelseregistret används också binär sökning. Programsteget måste vidare se till att jämförelsen mellan textavsnitt och lexikonord fortsätter på ett riktigt sätt, när lexikonordet består av nya delord efter ett trunkerat delord, dvs delordens exakta adress (fysiska placering) i ordföljderna måste kontrolleras och delorden jämföras var för sig. I denna kontroll utnyttjar vi den kvantifierade representationen av ord i lexikon, vilken tidigare beskrivits.

4.6 Utmatning

Subrutinen för sökning ger som resultat det identifierade lexikonordets ordningsnummer eller ett meddelande att ingen identifiering kunnat göras. I det första fallet skrivs textraden med fullständigt radnummer ut på en massminnesfil tillsammans med ordningsnumret. I det andra fallet skrivs textraden ut på en annan fil, den s k "slaskfilen". Den första filen, som innehåller ca 22 000 rader, sparas och används i den fortsatta datormaskinella bearbetningen, där lexikonorden får ersätta textraderna. Slaskfilen skrivs ut på radskrivare för en manuell kontroll.

4.7 Resursåtgång

I körningen tas olika delresurser hos datamaskinen i anspråk. Ett totalmått på körningens storlek är det pris som man får betala för körningen. Kostnaden för den slutliga körningen av sökprogrammet är ca 900 kronor. En enhetlig körning får inte vara väsentligt mycket större i omfattning om den skall få rimlig genomströmningstid och liten risk för att systemfel (med omkörning som följd) uppstår under körningens gång. Den stora tidsåtgången beror i hög grad på mångfalden av jämförelser mellan ord.

Programmet återges på sidorna 16-22. Som framgår ur texten finns det begränsningar som betingas av det material på vilket ANACONDA har tillämpats. Dessutom skulle kunna tänkas att ett annat programmeringsspråk kan hantera strängar mera effektivt än vad som är möjligt med NU-Algol-kompilatorn. Slutligen skall nämnas att vi hoppas kunna tillämpa ANACONDA på nya material. På så sätt hoppas vi kunna förbättra programmet och så småningom kunna få ett program som är både mera generellt och mera effektivt.

```
∇RUN LRBB,659174,LRBB,60,300
∇NUALG,IS HP
BEGIN
STRING ARRAY ADJ(36:1:586),VRB(36:1:1607),SBS(66:1:1634);
STRING ARRAY ADJBEG(2:1:190),VRBBEG(2:1:180),SBSBEG(2:1:250);
STRING ARRAY ADJEND(6:1 :41),VRBEND(6:1:37),SBSEND(6:1:77);
STRING ARRAY M(84:1:100);
INTEGER RLGD,NA,NV,NS,IEOF,MNR,MNR0,I,IMAX,J,NVE,JX,TS,KLASS,PKT,
LU1,LU2;
INTEGER BA,BV,BS;
INTEGER EA,EV,ES;
EXTERNAL FORTRAN PROCEDURE LAS,SKREOF,SKRIV;
STRING R(204),K(84),KS(2),K25(1);
INTEGER ARRAY RVE(1:20);
INTEGER ARRAY ADJADR,VRBADR,SBSADR(1:300);

PROCEDURE RLEX(LU,ORD,ORDLGD,NMAX,BIGBEG,BIGADR,BIGMAX);
VALUE LU,ORDLGD;
INTEGER LU,ORDLGD,NMAX,BIGMAX;
STRING ARRAY ORD,BIGBEG;
INTEGER ARRAY BIGADR;
BEGIN
EXTERNAL FORTRAN PROCEDURE LASLEX;
STRING RAD(84),BEG0(2),BEG(2);
INTEGER N,IEOF;
INTEGER BIGRAM;
N=0;
BEG0='  ';
BIGRAM=0;
L1:
LASLEX(LU,RAD,IEOF); IF IEOF EQL 1 THEN GO TO L2;
N=N+1;
ORD(1,ORDLGD:N)=RAD(1,ORDLGD);
BEG=RAD(5,2);
IF BEG NEQ BEG0 THEN BEGIN
BIGRAM=BIGRAM+1;
BIGADR(BIGRAM)=N;
BIGBEG(BIGRAM)=BEG;
BEG0=BEG END;
GO TO L1;
L2:
NMAX=N;
BIGMAX=BIGRAM;
BIGADR(BIGMAX+1)=NMAX+1
END;

PROCEDURE REND(ORDEND,NMAX);
INTEGER NMAX;
STRING ARRAY ORDEND;
BEGIN
```

DEKLARATIONER:
ADJEKTIV-,VERB- RESP SUBSTANTIVLEXIKON.
BEGYNNELSEBIGRAM FöR ADJEKTIV,VERB RESP SUBSTANTIV.
ADJEKTIV-, VERB- RESP SUBSTANTIVÄNDELSER.
LAGRINGSPLATS FöR EN MENING.
RÄKNARE.

EXTERNA PROCEDURER I FORTRAN.
BUFFERTAREOR.

ADR TILL FöRSTA LEXIKONORD MED GIVET BEGYNNELSEBIGRAM.

PROCEDUR RLEX:
INLÄSNING AV ETT LEXIKON. UPPBYGGNAD AV TABELLER MED BEGYNNELSEBIGRAM OCH MOTSVARANDE ADRESSER I LEXIKON.

PROCEDUR REND:
INLÄSNING AV ETT ÄNDELSELEXIKON.

```
INTEGER N;
STRING AEND(6);
N=0;
L1:
READ(<<A,S6>>,AEND,L2);
N=N+1;
ORDEND(N)=AEND;
GO TO L1;
L2:
NMAX=N;
END;
```

PROCEDUR LAGTXT:
LAGRA ORDENHET I BUFFERTEN R SÅ ATT FORTSÄTTNINGSRADER KOMMER MED.

```
PROCEDURE LAGTXT(I,TS); INTEGER I,TS;
BEGIN INTEGER J,TANT;
L1:
J=65;
L2:
IF M(J:I) EQL ' ' THEN BEGIN J=J-1;
IF J GTR 26 THEN GO TO L2 END;
TANT=J-26+1;
R(TS,TANT)=M(26,TANT:I);
TS=TS+TANT;
IF I NEQ IMAX AND M(9,2:I) EQL M(9,2:I+1) THEN
BEGIN I=I+1; R(TS)=' '; TS=TS+1; GO TO L1 END
END;
```

PROCEDUR SOEK:
I PROCEDUREN AVGÖRES OM BUFFERTEN R INNEHÅLLER ETT ORD I ETT GIVET LEXIKON.

```
PROCEDURE SOEK(ORD,BIGADR,BIGBEG,BIGMAX,ORDEND,ENDMAX,PKT);
VALUE BIGMAX,ENDMAX;
INTEGER BIGMAX,PKT;
INTEGER ENDMAX;
STRING ARRAY ORD,BIGBEG;
INTEGER ARRAY BIGADR;
STRING ARRAY ORDEND;
BEGIN
INTEGER J,MIN,MAX,LIM,JFR;
INTEGER VPIX,VPPKT;
INTEGER IXLGD;
STRING T(1);
```

PROCEDUR TEST (SUBRUTIN TILL PROCEDUR SOEK):
PROCEDUREN JÄMFÖR TEXTEN I R FR.O.M. KOL. J MED ORD NUMMER IX I LEXIKON.

A = AKTUELL STARTKOLUMN I LEXIKONORD.
I = AKTUELL STARTKOLUMN FÖR TEXTAVSNITT.
JÄMFÖRELSE AV NYTT DELORD I LEXIKONORDET MED TEXTEN:
M = LÄNGD PÅ AKTUELLT DELORD I LEXIKONORD.
L = ANTAL TECKEN SOM JÄMFÖRES.
JFR = JÄMFÖRELSENS UTFALL. 0=LIKHET. -1,+1=OLIKHET.

```
PROCEDURE TEST(IX); VALUE IX; INTEGER IX;
BEGIN INTEGER M,I,A,L;
STRING AEND(6);
INTEGER K,KB;
INTEGER LIM,MIN,MAX;
A=1;
I=J;
L1:
M=ORD(A,2:IX);
L=IF M+I LEQ RLGD+1 THEN M ELSE RLGD+1-I;
A=A+4;
JFR=IF R(I,L) EQL ORD(A   ,L:IX) THEN 0 ELSE IF R(I,L) LSS ORD(A   ,L:IX)
```

```
THEN -1 ELSE +1;
IF JFR EQL 0 THEN KB=I+L-1;
```

KB = STARTKOLUMN FÖR EV. ÄNDELSE I TEXTEN.
OM LIKHET HITTILLS GÄLLER OCH DELORDET I LEXIKON ÄR TRUNKERAT SA TAGES ÄNDELSEN I TEXTEN UT OCH LAGRAS I AEND.

```
IF JFR EQL 0 AND ORD(A-2:IX) EQL 0 THEN BEGIN
IF KB EQL RLGD THEN GO TO L4;
K=0; AEND='      ';
L3: KB=KB+1; IF KB LEQ RLGD THEN BEGIN
T=R(KB); IF T NEQ ' ' AND T NEQ '(' AND T NEQ ')' THEN BEGIN
K=K+1;
IF K LEQ 6 THEN BEGIN AEND(K)=T; GO TO L3 END ELSE
BEGIN JFR=-1; GO TO L4 END
END
END;
KB=KB-1;
IF K EQL 0 THEN GO TO L4;
```

FINN ÄNDELSE I ÄNDELSEREGISTRET SOM ÄR LIKA MED AEND.
METOD: BINÄR SÖKNING.

```
MIN=1;
MAX=ENDMAX;
IF AEND LSS ORDEND(MIN) THEN BEGIN JFR=-1; GO TO L4 END;
IF AEND EQL ORDEND(MIN) THEN GO TO L4;
IF AEND GTR ORDEND(MAX) THEN BEGIN JFR=-1; GO TO L4 END;
IF AEND EQL ORDEND(MAX) THEN GO TO L4;
L5:
```

OM AEND EJ FINNS I ÄNDELSEREGISTRET SÄTTS JFR=-1.

```
IF MIN+1 EQL MAX THEN BEGIN JFR=-1; GO TO L4 END;
LIM=(MIN+MAX)//2;
IF AEND EQL ORDEND(LIM) THEN GO TO L4 ELSE
IF AEND LSS ORDEND(LIM) THEN MAX=LIM ELSE MIN=LIM;
GO TO L5;
END;
```

OM LIKHET HITTILLS GÄLLER OCH LEXIKONORDET FORTSÄTTER MED NYTT DELORD INITIERAS JÄMFÖRELSE AV DETTA MED FORTSÄTTNING I TEXTEN OCH ÅTERHOPP TILL L1.

```
L4:
IF JFR EQL 0 AND ORD(A-1:IX)    EQL 1 THEN BEGIN
I=I+M;
L2:
IF I LSS RLGD THEN BEGIN
T=R(I);
IF T EQL '(' OR T EQL ')' THEN BEGIN JFR=-1; GO TO L6 END ELSE
IF T NEQ ' ' THEN BEGIN I=I+1; GO TO L2 END;
I=I+1; A=A+M; GO TO L1;
END;
END;
```

OM LIKHET HITTILLS GÄLLER SÄTTS OLIKHET OM SISTA ORDET I TEXTRADEN ÄR KORTARE ÄN DET ORD SOM DET JÄMFÖRTS MED ELLER OM SISTA ORDET I LEXIKONORDET EJ ÄR TRUNKERAT OCH KORTARE ÄN DET ORD SOM DET JÄMFÖRTS MED.
IXLGD = SLUTKOLUMN FÖR DET IDENTIFIERADE TEXTAVSNITTET.

```
IF JFR EQL 0 AND M+I GTR RLGD+1 THEN JFR=-1;
IF JFR EQL 0 AND ORD(A-2,1:IX) EQL 1 AND I+L NEQ RLGD+1 THEN BEGIN
T=R(I+L); IF T  NEQ ' ' AND T NEQ '(' AND T NEQ ')' THEN JFR=-1 END;
IF JFR EQL 0 THEN IXLGD=KB;
L6:
END;
```

PROCEDUREN SOEK STARTAR. J = STARTKOLUMN FÖR JÄMFÖRELSEN.
VPIX = ANGER OM TEXTDELEN ÄR INOM PARENTES ELLER EJ.
VPPKT= ANGER OM REDAN IDENTIFIERAT ORD ÄR INOM PARENTES.
IDENTIFIERING AV TEXTDEL SOM INLEDS MED NYTT ORD BÖRJAR.

```
J=26;
VPIX=0;
PKT=0; VPPKT=0;
L1: IF J GTR RLGD THEN GO TO L5;
T=R(J);
IF T EQL ' ' THEN BEGIN J=J+1; GO TO L1 END;
```

```
IF T EQL '(' THEN BEGIN J=J+1; VPIX=1; GO TO L1 END;
MIN=1;
MAX=BIGMAX;
IF R(J,2) LSS BIGBEG(MIN) THEN GO TO L4;
IF R(J,2) EQL BIGBEG(MIN) THEN BEGIN LIM=MIN; GO TO L3 END;
IF R(J,2) GTR BIGBEG(MAX) THEN GO TO L4;
IF R(J,2) EQL BIGBEG(MAX) THEN BEGIN LIM=MAX; GO TO L3 END;
L2:
IF MIN+1 EQL MAX THEN GO TO L4;
LIM=(MIN+MAX)//2;
IF R(J,2) EQL BIGBEG(LIM) THEN GO TO L3 ELSE
IF R(J,2) LSS BIGBEG(LIM) THEN MAX=LIM ELSE MIN=LIM;
GO TO L2;
L3:
MIN=BIGADR(LIM); MAX=BIGADR(LIM+1)-1;
FOR LIM=(MIN,1,MAX) DO BEGIN
TEST(LIM);
IF JFR EQL 0 THEN BEGIN
IF NOT (VPPKT EQL 1 AND VPIX EQL 0) THEN BEGIN PKT=LIM; VPPKT=VPIX END
END END;
IF PKT NEQ 0 AND J LSS IXLGD THEN J=IXLGD;
L4: J=J+1; IF J GTR RLGD THEN GO TO L5;
T=R(J); IF T NEQ ' ' AND T NEQ '(' AND T NEQ ')' THEN GO TO L4;
IF T EQL '(' THEN VPIX=1 ELSE IF T EQL ')' THEN VPIX=0;
J=J+1; GO TO L1;
L5:
END;
```

FINN SEGMENT I LEXIKON DÄR ORDEN BÖRJAR MED SAMMA TVÅ BOKSTÄVER (BIGRAM) SOM TEXTDELEN. METOD: BINÄR SÖKNING I BIGRAM-TABELL.

MIN = STARTADRESS I SEGMENTET. MAX = SLUTADRESS. JÄMFÖR ORD EFTER ORD I SEGMENTET MED TEXTDELEN.

JFR = 0 OM TEXTDELEN IDENTIFIERATS MED LEXIKONORDET. DET IDENTIFIERADE ORDETS NUMMER ANGES I PKT SAVIDA INTE ETT TIDIGARE IDENTIFIERAT ORD INOM PARENTES PRIORITERAS. INITIERING AV JÄMFÖRELSE AV NÄSTA TEXTAVSNITT I R OCH ATERHOPP TILL L1 OM RADEN INTE ÄR SLUT.

```
PROCEDURE UTFALL(ORD,PKT); VALUE PKT; STRING ARRAY ORD; INTEGER PKT;
BEGIN
INTEGER A,M;
FOR TS=(RLGD+1,1,80) DO R(TS)=' ';
R(81)=J; R(82)=' ';
R(83)=KLASS;
WRITE(CORE(R(84,4)),<<I4,A1>>,PKT);
TS=88;
A=1;
L1:
M=ORD(A,2:PKT);
A=A+4;
R(TS,M)=ORD(A,M:PKT); TS=TS+M;
IF ORD(A-2:PKT) EQL 0 THEN BEGIN R(TS)='*'; TS=TS+1 END;
IF ORD(A-1:PKT) NEQ 0 THEN BEGIN A=A+M; R(TS)=' '; TS=TS+1;
GO TO L1 END;
L2: IF TS LEQ 120 THEN BEGIN R(TS)=' '; TS=TS+1; GO TO L2 END;
SKRIV(LU1,R,20)
END;
```

PROCEDUR UTFALL:
UTSKRIFT AV POST PA MASSMINNESFIL FÖR TEXTRAD SOM IDENTIFIERATS I LEXIKON. INNEHALL:

```
I KOL   1-23:TEXTRADENS IDENTIFIERARE
       24-25:SATSKOD
       26-80:TEXTEN(EV. AVKORTAD)
          81:HUVUDSATS-BISATS-NUMMER (1-5)
          83:KLASS (ADJEKTIV, VERB ELLER SUBSTANTIV)
       84-87:LEXIKONORDETS IDENTIFIERARE
      88-120:LEXIKONORDET I KLARTEXT.
```

```
PROCEDURE EJUTF;
BEGIN
FOR TS=(RLGD+1,1,84) DO R(TS)=' ';
SKRIV(LU2,R,14)
END;

RLEX(11,ADJ,36,NA,ADJBEG,ADJADR,BA);
WRITE('ADJ-BIGRAM=',BA);
RLEX(12,VRB,36,NV,VRBBEG,VRBADR,BV);  WRITE('VRB-BIGRAM=',BV);
RLEX(13,SBS,66,NS,SBSBEG,SBSADR,BS);  WRITE('SBS-BIGRAM=',BS);
REND(ADJEND,EA); WRITE('ANTAL ADJEKTIVÄNDELSER=  ',EA);
REND(VRBEND,EV); WRITE('ANTAL VERBÄNDELSER=      ',EV);
REND(SBSEND,ES); WRITE('ANTAL SUBSTANTIVÄNDELSER=',ES);
LU1=9;
LU2=14;
LAS(K,IEOF); IF IEOF EQL 1 THEN BEGIN WRITE('FEL. INGA INDATA');
GO TO SLUT END;
L0:
MNR0=K(1,8);
I=0;
L1:
I=I+1; M(I)=K;
LAS(K,IEOF); IF IEOF EQL 1 THEN GO TO L2;
MNR=K(1,8);
IF MNR EQL MNR0 THEN GO TO L1;
L2:
IMAX=I;
FOR J=(1,1,5) DO BEGIN
NVE=0;
JX=66+(J-1)*3;
FOR I=(1,1,IMAX) DO BEGIN
KS=M(JX,2:I); IF KS EQL '  ' THEN GO TO L3;
IF KS EQL '40' THEN BEGIN
K25=M(25:I); IF K25 EQL ' ' OR K25 EQL '1' THEN BEGIN
NVE=NVE+1; RVE(NVE)=I END END
ELSE BEGIN
KLASS=IF KS EQL '32' OR KS EQL '52' OR KS EQL '72' THEN 1 ELSE
IF KS EQL '30' OR KS EQL '33' OR KS EQL '50' OR KS EQL '53' OR
   KS EQL '70' OR KS EQL '73' OR KS EQL '44' OR KS EQL '45' OR
   KS EQL '46' THEN 3 ELSE 0;
IF KLASS NEQ 0 THEN BEGIN
R(1,23)=M(1,23:I); R(24,2)=KS;
TS=26; LAGTXT(I,TS); RLGD=TS-1;
IF KLASS EQL 1 THEN BEGIN
SOEK(ADJ,ADJADR,ADJBEG,BA,ADJEND,EA,PKT);
IF PKT NEQ 0 THEN UTFALL(ADJ,PKT) ELSE EJUTF END ELSE
BEGIN
SOEK(SBS,SBSADR,SBSBEG,BS,SBSEND,ES,PKT);
IF PKT NEQ 0 THEN UTFALL(SBS,PKT) ELSE EJUTF END
END;
```

PROCEDUR EJUTF:
UTSKRIFT AV POST PÅ MASSMINNESFIL FÖR TEXTRAD SOM EJ IDENTIFIERATS I LEXIKON. INNEHÅLL ÄR LIKA MED KOL 1-80 OVAN.

HUVUDPROGRAMMET BÖRJAR:
INLÄSNING AV ADJEKTIV-, VERB-, SUBSTANTIVLEXIKON. BILDA TABELLER MED DE OLIKA BEGYNNELSEBIGRAM SOM FINNS, SAMT ADRESSER TILL MOTSVARANDE FÖRSTA ORD I LEXIKON.
INLÄSNING AV ÄNDELSER.

UTFILERNA FÖR IDENTIFIERADE RESP EJ IDENTIFIERADE ORD GES SINA ENHETSNUMMER.
INITIERING AV LÄSNING AV TEXTFILEN.

INLÄSNING AV EN MENING I TEXTFILEN.

EFTER HAND GENOMGÅNG AV HUVUDSATS OCH FYRA BISATSER:

EFTER HAND GENOMGÅNG AV ALLA RADER I MENINGEN:

OM SATSKODEN = 40(VERB), ÖKA RÄKNAREN NVE MED 1 OCH REGISTERA RADENS NUMMER I FÄLTET RVE;

REGISTRERA ANNARS OM RADEN KLASSIFICERATS SOM ADJEKTIV ELLER SUBSTANTIV.

OM RADEN ÄR ADJEKTIV ELLER SUBSTANTIV SÅ LAGRA DESS IDENTITETSBEGREPP I BUFFERTEN R: KOL 1-23, SATSKOD I KOL 24-25 OCH TEXTEN FR.O.M. KOL 26. TEXT PÅ FORTSÄTTNINGSRAD MEDTAGES. SLUTKOLUMNEN LAGRAS I RLGD. TEXTRADEN JÄMFÖRES MED ADJEKTIV- RESP. SUBSTANTIVLEXIKON I PROCEDUREN SOEK. OM ETT MOTSVARANDE LEXIKONORD FINNS LAGRAS DESS NUMMER I PKT SOM ANNARS BLIR 0. UTFALLET REGISTRERAS PÅ FILEN FÖR ÅTERVUNNA ORD ELLER SLASKFILEN.

```
END;
L3:
END;
IF NVE NEQ 0 THEN BEGIN
I=RVE(1);
R(1,23)=M(1,23:I); R(24,2)='40';
TS=26; LAGTXT(I,TS);
FOR JX=(2,1,NVE) DO BEGIN
R(TS)=' '; TS=TS+1;
I=RVE(JX); LAGTXT(I,TS) END;
RLGD=TS-1;
KLASS=2;
```

OM VERB FINNS I SATSEN SÅ SKER EN LAGRING I BUFFERTEN R. OM FLERA VERB FINNS SÅ KOMBINERAS DE TILL EN ORD-ENHET I R.

```
SOEK(VRB,VRBADR,VRBBEG,BV,VRBEND,EV,PKT);
IF PKT NEQ 0 THEN UTFALL(VRB,PKT) ELSE EJUTF END;
END;
```

VERBET JÄMFÖRES MED VERBLEXIKON.

```
IF IEOF EQL 0 THEN GO TO L0;
SKREOF(LU1,LU2);
SLUT:
END
```

ÅTERHOPP TILL LÄSNING AV NY MENING.
EOF-MÄRKE SÄTTS PÅ UTFILERNA.

```
∇FOR,IS S1
      SUBROUTINE SKRIV(LU,R,N)
      DIMENSION R(34)
      WRITE(LU,101)(R(I),I=1,N)
  101 FORMAT(20A6)
      RETURN
      END
```

FORTRAN-SUBRUTIN: SKRIVER EN RAD PÅ 120 TECKEN.

```
∇FOR,IS S2
      SUBROUTINE SKREOF(LU1,LU2)
      END FILE LU1
      END FILE LU2
      RETURN
      END
```

FORTRAN-SUBRUTIN: SÄTTER EOF-MÄRKE PÅ TVÅ FILER.

```
∇FOR,IS S3
      SUBROUTINE LAS(K,IEOF)
      DIMENSION K(14)
      READ(8,101,END=1)K
  101 FORMAT(13A6,A2)
      IEOF=0
      RETURN
    1 IEOF=1
      RETURN
      END
```

FORTRAN-SUBRUTIN: LÄSER EN KORTBILD FRÅN ENHET 8.

```
∇FOR,IS S4
      SUBROUTINE LASLEX(LU,RAD,IEOF)
      DIMENSION RAD(14)
      READ(LU,101,END=1)RAD
  101 FORMAT(13A6,A2)
      IEOF=0
      RETURN
    1 IEOF=1
```

FORTRAN-SUBRUTIN: LÄSER EN KORTBILD FRÅN ENHET LU.

```
        RETURN
        END
 ∇ASG,AX IP01-40              TILLORDNING AV HUVUDFILEN MED TEXT.
 ∇USE 8,IP01-40               FILEN GES ALTERNATIVT NAMN FØR HANTERING I FORTRAN.
 ∇DELETE,C ATERVUNN.
 ∇ASG,U ATERVUNN,F40///512    TILLORDNING AV SKRIV-FIL FØR IDENTIFIERADE ORD.
 ∇USE 9,ATERVUNN
 ∇ASG,U SLASK-ORD             TILLORDNING AV SKRIVFIL FØR EJ IDENTIFIERADE ORD.
 ∇USE 14,SLASK-ORD
 ∇ASG,AX TRUNK-A              TILLORDNING AV ADJEKTIV-LEXIKON.
 ∇ASG,AX TRUNK-V              TILLORDNING AV VERB-LEXIKON.
 ∇ASG,AX SUBST-LEX            TILLORDNING AV SUBSTANTIVLEXIKON.
 ∇USE 11,TRUNK-A
 ∇USE 12,TRUNK-V
 ∇USE 13,SUBST-LEX
 ∇ASG,A ADJEND                TILLORDNING AV ADJEKTIV-ÄNDELSER.
 ∇ASG,A VRBEND                TILLORDNING AV VERB-ÄNDELSER.
 ∇ASG,A SBSEND                TILLORDNING AV SUBSTANTIV-ÄNDELSER.
 ∇XQT                         STARTA KØRNING AV PROGRAMMET.
 ∇ADD ADJEND.                 FILERNA MED ÄNDELSER LÄGGS IN SOM KORTBILDER EFTER
 ∇EOF                         ∇XQT-KORTET SA ATT DE I ALGOL-PROGRAMMET LOGISKT
 ∇ADD VRBEND.                 LÄSES FRAN KORTLÄSAREN.
 ∇EOF
 ∇ADD SBSEND.
 ∇EOF
 ∇DATA,L SLASK-ORD            KONTROLLUTSKRIFT AV FILEN SLASK-ORD EFTER KØRNING.
 ∇END
LINES:  328

∇ FIN
```

5. BILAGA

DATORBASERAD INNEHÅLLSANALYS: ANACONDA

5. BILAGA

DATORBASERAD INNEHÅLLSANALYS: ANACONDA

Inom beteendevetenskaplig forskning har använts och används många olika innehållsanalystekniker, vilket innebär att beteendevetare kan anses vara väl förtrogna med såväl de teoretiska som de metodiska och tekniska eller praktiska problem som användningen av klassiska innehållsanalystekniker medför. En vetenskapligt genomförd innehållsanalys innebär att forskaren, oberoende av ett visst bestämt resultat, måste kunna redogöra för den valda metodens reliabilitet och validitet. En omsorgsfull, objektiv, reliabel och valid analys av text är emellertid mycket tidskrävande, och omfattande textanalyser kräver utveckling av mekaniserade eller automatiserade rutiner. I och med att datorer kan användas för ihågkommande och logisk selektion, hoppas vi kunna utveckla en datorbaserad innehållsanalysmetod som kännetecknas av större objektivitet och flexibilitet än vad som är fallet hos de klassiska innehållsanalysteknikerna.

Syftet med den här översikten är (1) att underlätta för läsaren förståelsen av vår metod för en analys av concept genom datorbearbetning (ANACONDA) och (2) att presentera ledtrådar, som vi hoppas ska ge en tillräckligt bra bild av de enskilda stegen i den kumulativa processen att utveckla en teori och en metod för att studera komplexa psykologiska fenomen, som t ex att meddela sig via symboler.

5.1 Problemställning

Att utveckla idéer, formulera och lösa problem är beteenden som är intimt förknippade med människans specifika förmåga att uttrycka sig verbalt. Försök att explorera och studera inte bara problemlösningsbeteenden utan även de komplexa psykologiska processerna bakom sådana beteenden sker inte bara inom psykologin utan sedan ett 20-tal år tillbaka inom flera andra vetenskapsgrenar, t ex matematik, artificiell intelligens, informationsbehandling och kvantitativ lingvistik. Det som förenar forskningsinsatserna inom de olika disciplinerna är målsättningen att utröna och göra synliga osynliga symboliska processer och mekanismer. Istället för att med hjälp av enkla matematiska formler försöka beskriva sådana kognitiva operationer, försöker forskare inom de nämnda områdena att utveckla datorprogram för att beskriva dessa och testa modeller och teorier om hur komplexa psykologiska strukturer ser ut och hur processer utvecklar sig.

5.2 Innehålls- och kontextorienterade teorier

Syftet med att försöka utveckla en teori om innebörden i meddelanden som människor kommunicerar sinsemellan är att öka vår förståelse av de kognitiva struk-

turer som antas ligga till grund för människans verbala uttryck. Genom att steg för steg utveckla och kontinuerligt utpröva våra antaganden försöker vi bestämma deras hållbarhet. Den fråga som vi har ställt oss är denna:

KAN VI MED HJÄLP AV NUMERISK ANALYS OCH GENOM KVANTITATIV BESTÄMNING IDENTIFIERA OCH KATEGORISERA KOGNITIVA STRUKTURER I VERBALA DATA, T EX INTERVJUTEXTER?

Mot bakgrund av de sista årens vetenskapliga debatt om processforskning och de påtagliga begränsningar som olika skattningsscheman som datainsamlingsmetod har, borde ett framgångsrikt genomförande av den uppgift vi angripit vara ett betydelsefullt bidrag till de forskningsmetoder som står till buds för samhällsvetare.

5.3 Guide till metodutvecklingen

Generellt om metod och modell. Skriven eller talad text kännetecknas av stor komplexitet och av att den typ av information som skall utvinnas ur ett material sällan eller aldrig är samlad på ett enda ställe i texten. Om strukturella relationer ändå skall kunna framträda krävs det att man preparerar texten på basis av vissa grundantaganden. I Bierschenk & Bierschenk (1976) presenteras ett flödesschema som anger de enskilda stegen i utvecklingen av den metod för datorbaserad innehållsanalys som vi föreslår. Dessutom presenteras den psykolingvistiska modell som ligger till grund för vår analystext.

Nedan presenterar de olika faserna i kronologisk ordning.

Grundmaterial. Vi har avgränsat vårt empiriska material till att tills vidare gälla "Informationssökning, dokumentation och forskningsplanering för skolans F&U-arbete". Uppläggningen och genomförandet av en intervjustudie kring detta redovisas i B. Bierschenk (1974 a). I denna rapport utvärderas också skattningsskalor som intervjupersonerna konfronterades med under intervjun.

Impressionistisk analys. Ett sätt att utvärdera intervjutexter är att använda en impressionistisk innehållsanalys. Den bygger på intuition, inlevelse och intryck, som innebär att tolkningen bygger på subjektivt funna analysresultat. En sådan analys föreligger i Annerblom (1974).

Datorbaserad analys. En modell för sökning efter information i intervjutexter, en kortfattad beskrivning av preliminära kodningsregler och några empiriska resultat från utprövning av manuell tilldelning av koder på intervjutext presenteras i B. Bierschenk (1974 b).

Konstruktion av ett regelsystem. Förutsättningarna för att redan utvecklade metoder skall kunna tillämpas är ofta knutna till materialets utseende. De analysförsök som redovisats i litteraturen och som är av intresse för vår analys är utvecklade med skriven text som bas. Men eftersom vårt material är talspråkstext (utskriven från ljudband) som när den yttrades endast var avsedd för en person, intervjuaren, blev det nödvändigt att bygga upp ett eget system av regler och koder. En preliminärmanual och några utprövningsresultat presenteras i I. Bierschenk (1974).

Tillförlitlighet i manuell kodning. Genom begreppet "datorbaserad innehållsana-

lys" vill vi markera att vi inte avser att utveckla en metod för automatisk textanalys. Detta betyder samtidigt att grundmaterialet först måste kodas innan maskinella bearbetningar av olika slag kan ta vid. Med vilken framgång två av varandra oberoende kodare har kunnat tillämpa olika kodningsregler på ett likartat sätt redovisas i Berg (1974).

Representation av manifesta språkstrukturer. Datorbaserad innehållsanalys börjar bli mer och mer använt och användbart internationellt. Kravet på program och tekniker som är avpassade för olika problemområden gör sig mer och mer gällande. Den teori för språklig representation som vi funnit mest intressant för vår analys är Schank's "Conceptual Dependency Theory". Vi kom i kontakt med den efter att våra första kodningsregler utarbetats, och vi fann att den ligger i linje med vårt sätt att behandla texten för input i dator. På vilket sätt inmatning sker, hur identifikation specificeras i kodningen och på vilket sätt vi bygger upp vår lexikonbas redovisas i I. Bierschenk (1975).

Teoretiska och psykometriska problem. Att försöka kartlägga vad som i forskningslitteraturen egentligen menas med en innehållsanalys är mycket svårt. Till grund för varje innehållsanalysteknik ligger nämligen ett specifikt sätt att betrakta innehållet i ett meddelande. En innehållsanalys förutsätter att vi kan definiera analysens mätobjekt, dvs det som skall mätas och räknas. Analysens enhet i vår psykolingvistiska modell går tillbaka på det välkända paradigmet Agent-aktion-Objekt (mål). Komponenterna definieras kortfattat nedan:

Analysens paradigm 1. Agent. Aktionscentrum eller målsökande entitet som använder sig av resurser för att realisera sina mål. Denna definition innefattar också grupper, organisationer eller abstraktioner som fyller funktionen av att vara agent. 2. Aktion. En riktad handling som utförs av en agent i syfte att realisera mål. Handlingen definierar innebörden i AaO-paradigmet. 3. Objekt. Allt som en handling kan vara riktad mot eller utföras med.

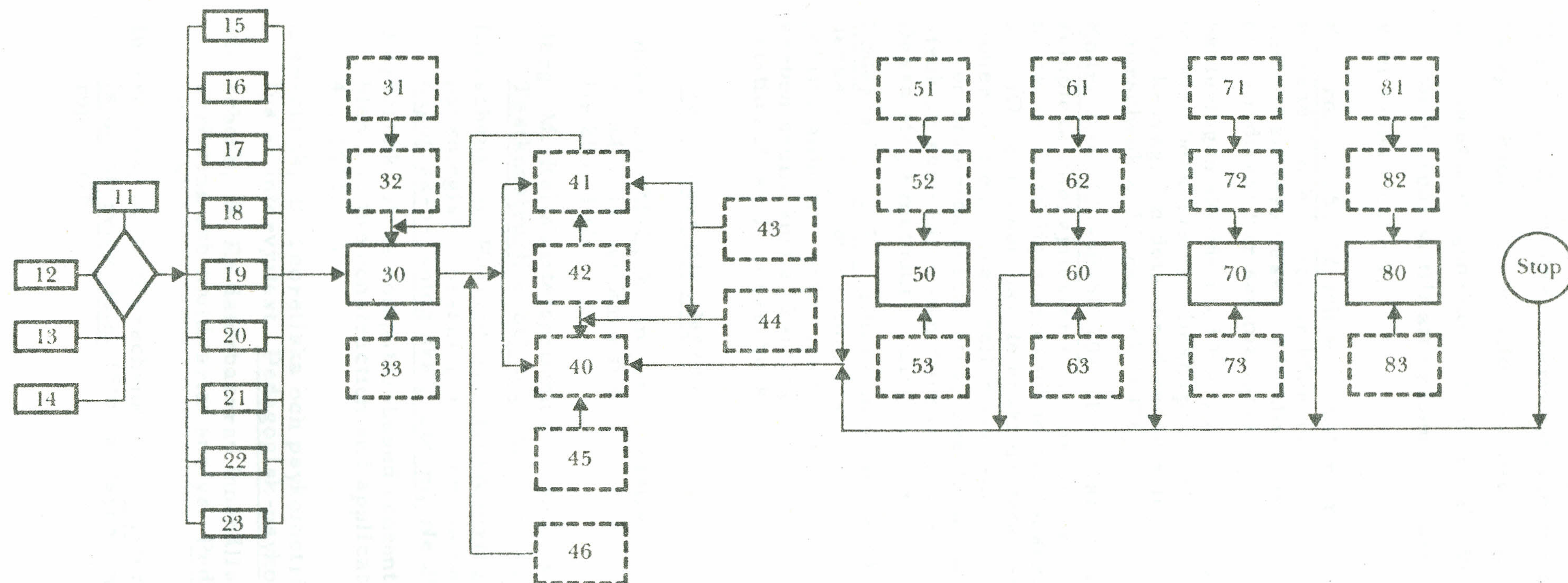
Genom AaO-paradigmet avgränsas de komponenter som utgör ett naturligt sammanhang, dvs en observerbar sats. Med detta avses den grundläggande formen för ett påstående som uttrycks genom $substantiv_1$-verb-$substantiv_2$-relationen. Medan agent och objekt (substantiv) specificeras genom attribut (adjektivfraser), anges genom verbet relationen mellan substantiven, dvs handlingar, händelser eller tillstånd. Ordningen mellan dessa grundelement anges genom syntax. Genom ett lexikon och ett regelsystem hoppas vi kunna konstruera teorier och modeller som kan användas för att beskriva händelser.

Med hjälp av de lingvistiska element som finns i flödesschemat i figur 1 skall vi visa dels hur vi bygger upp concept i en given kontext, dels på vilket sätt vi avser att numeriskt beskriva texten och göra kvantitativa analyser. Hur vi utnyttjar dessa möjligheter beskrivs i B. Bierschenk (1976).

5.4 Numerisk beskrivning och kvantitativa analyser

Att bygga upp concept i en given kontext förutsätter ett regelsystem som anger hur olika element skall kopplas med varandra och i vilken ordning detta skall ske. Vi antar t ex att de relationer som finns mellan substantiv och adjektiv och mellan substantiv och verb återspeglar sådana relationer som kopplar empiriska fenomen med varandra. Om vi t ex vill ange att en uppsättning substantiv modifieras av en uppsättning adjektiv så kan vi formalisera denna relation.

Vidare antar vi att substantiv får sitt empiriska innehåll genom adjektiv och/eller verb som de är kopplade med. Genom att skalera adjektiv och verb



11 Informationskälla
12 Negation
13 Tid
14 Modalitet
15 Villkor
16 Orsak
17 Medgivande
18 Avsikt
19 Disjunktion
20 Komparation
21 Interrogation
22 Antagande
23 Önskan
30 Agent
31 Bestämning
32 Beskrivning
33 Klassificering
40 Händelse/tillstånd
41 Kopula
42 Satsmodifierare
43 Tidsbestämning
44 Platsbestämning
45 Handlings-/ tillståndsmodifierare
46 Omständighet
50 Objekt
51 Bestämning
52 Beskrivning
53 Klassificering
60 Riktning/lokalisering
61 Bestämning
62 Beskrivning
63 Klassificering
70 Mottagare
71 Bestämning
72 Beskrivning
73 Klassificering
80 Instrument
81 Bestämning
82 Beskrivning
83 Klassificering

Figur 1. Programflödesplan för en analys av text och uppbyggande av concept

kan vi åstadkomma numeriska beskrivningar och kvantitativa analyser av text. När vi på så sätt har observerat likheter eller kovariationer och definierat olika egenskaper, kan vi utföra multivariata analyser för att bestämma positionen av en bestämd egenskap i ett antal latenta dimensioner.

Nu övergår vi till att presentera enskilda analyser utifrån den kodstruktur som anges i figur 1.

Koderna (32, 52, 72 och 40). Eftersom vi i vår analys tar hänsyn till "syntaktiskt beteende" och betraktar både adjektiv och verb som beskrivande concept, har vi valt att skalera dessa. Adjektiv beskriver ett substantiv direkt, medan verben mera indirekt har samma funktion.

Skaleringen skedde med hjälp av sjugradiga skattningsskalor, vars bipolära ändpunkter beskrivs av adjektivparen (1) positiv-negativ, (2) aktiv-passiv och (3) stark-svag. En detaljerad redovisning av detta finns i B. Bierschenk (1976) och Bierschenk & Bierschenk (1976).

Kodrelation (30-40-50/70). En strategi för att övervinna de begränsningar som komplexa intervjutexter och stora datamängder medför är att bryta ned dessa i block av hanterlig storleksordning. Baserat på enkla numeriska beskrivningar av AaO-relationen har de enskilda dataelementen kondenserats. De logiska relationer som existerar mellan koderna 30-40-50(70) har utnyttjats för att ge objekten en empirisk bestämning, dvs de omvandlas till concept. På basis av en grundstruktur studeras sedan dimensionaliteten med hjälp av en diskriminantanalys. Dessa steg i metodutvecklingen beskrivs i detalj i B. Bierschenk (1976 b).

Koderna 32, 52, 72. Eftersom vi antar att adjektiv beskriver substantiven genom deras värde på en skattningsskala, har vi studerat de adverb som modifierar respektive adjektiv. Syftet med denna studie var att utveckla en princip för hur adverben skulle kunna skaleras. En principdiskussion kring adverbens tänjande funktion förs i I. Bierschenk (1977).

5.5 Referenser till ANACONDA

Annerblom, M-L. En impressionistisk innehållsanalys av intervjuer med forskare på pedagogiska institutioner i Sverige. Pedagogisk-psykologiska problem Nr 255, 1974.

Berg, M. Reliabilitetsprövning av en metod för innehållsanalys av intervjutext. Testkonstruktion och testdata, Nr 26, 1974.

Bierschenk, B. Perception, strukturering och precisering av pedagogiska och psykologiska forskningsproblem på pedagogiska institutioner i Sverige. Pedagogisk-psykologiska problem, Ne 254, 1974 (a).

Bierschenk, B. A computer-based content analysis of interview data: Some problems in the construction and application of coding rules. Didakometry, No. 45, 1974. (b)

Bierschenk, B. Teoretiska och psykometriska problem vid en datorbaserad analys av intervjutext. Pedagogisk-psykologiska problem, Nr 287, 1976.

Bierschenk, B. En datorbaserad innehållsanalys av intervjutext: Numerisk beskrivning och multivariat analys. Pedagogisk-psykologiska problem, Nr 307, 1976 (b).

Bierschenk, B. & Bierschenk, I. A system for a computer-based content analyis of interview data. (Studia Psychologica et Paedagogica, 32.) Lund: Gleerup, 1976.

Bierschenk, I. Konstruktion av ett regelsystem för en datorbaserad innehållsanalys av intervjutext: Preliminärmanual och några utprövningsresultat. Testkonstruktion och testdata, Nr 25, 1974.

Bierschenk, I. Datorbaserad innehållsanalys: Teoretiska och praktiska överväganden. Pedagogisk-psykologiska problem, Nr 283, 1975.

Bierschenk, I. En studie av modifierande ords kontextuella funktion: En principdiskussion och några kvantifieringsförsök. Pedagogisk-psykologiska problem, Nr 314, 1977.

ISSN 0346-5004